2011

Fostering Logical Thinking in Novice Student Programmers

Kyle Wenholz
University of Puget Sound, kwenholz@pugetsound.edu

Fostering Logical Thinking in Novice Programmers

Kyle Wenholz, advised by Professor David Akers

Motivation

Early in their education, student programmers learn to debug their programs, and some students debug by tinkering[1]. Tinkering is a rapid and haphazard coding style that makes almost any success a product of sheer chance. While quick exploration of code and solutions can be very helpful for experts, the student programmer is still learning to explore thoughtfully.

In the example below, a student struggles to correct syntactically broken code.

```
public MyClass(int i int j)
')' expected
```

"Maybe I just need one more parenthesis."

```
public MyClass(int i int j))
```

"Maybe I need to enclose my variables."

```
public MyClass((int i) (int j))
```

"Maybe I should separately parenthesize my parameters."

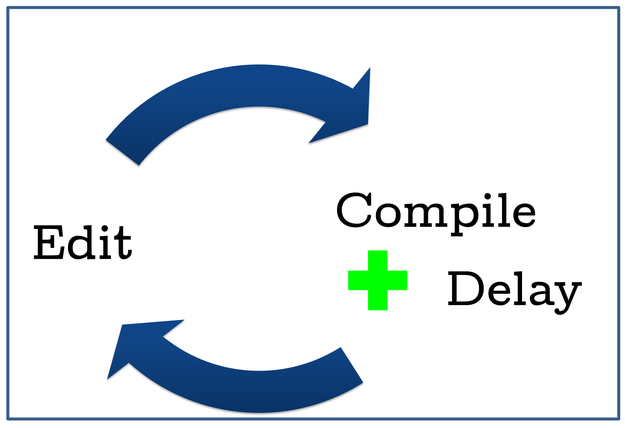
```
public MyClass(int i) (int j)
```

Acknowledgements

The project was funded by a McCormick Summer Research Grant. Thank you to my advisor, Professor David Akers.

Proposed Solution

Creating an artificial delay in the compilation step should change the cost structure of debugging by forcing students to think more before compiling. This, in turn, should disrupt tinkering. We hypothesized that students experiencing the delay would use the internet or books for help and write (trace code, plan tests, etc.) more often. While we expected students to compile less, we hoped that a smaller percentage of these compiles would produce errors.



Experimental Design

We ran a pilot study with 10 participants, having each attempt six short debugging tasks in the Java language: three with a compile delay and three without. This within-subjects experiment allowed participants to experience both conditions (counterbalanced between subjects) and allowed them to compare and contrast the conditions.

Measurements

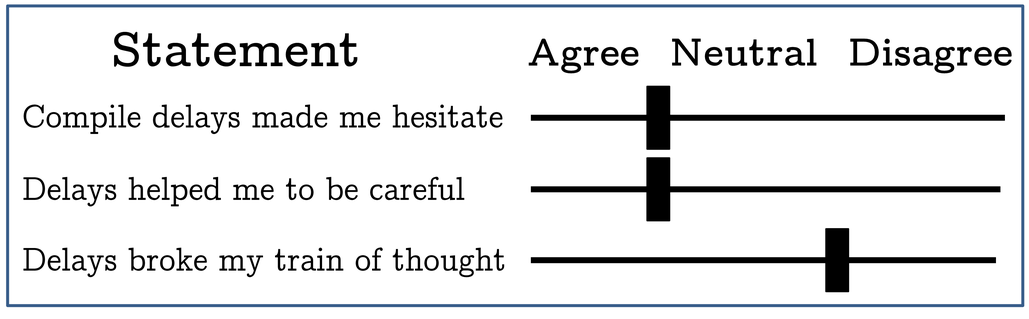
A researcher sat in on every experimental session to observe behavior, and we used the BlueJ programming environment to implement a plugin for collecting data related mainly to compiles. We recorded screen capture video and audio of each session, asking participants to think aloud as they worked.

Quantitative and Qualitative Data

- Total number of compiles
- Number of successful compiles
- Tasks completed
- Tasks skipped
- Frequency of internet usage
- Frequency of writing
- Observations
- Semi-structured interview
- Post-questionnaire
- Think-aloud audio
- Screen capture video

Results

Lacking statistically significant quantitative data, we focused mainly on the qualitative data gathered. Participants generally found the delay frustrating, but most also commented that they behaved more cautiously or were hesitant to compile with the delay present.



Future Work

This fall we plan to prototype a more active intervention system based on questions that guide the student to form stronger hypotheses about their actions. We may ask students to record hypotheses at compile time or lead them through this process using a dialog tree.

References:
[1] D.N. Perkins, C. Hancock, R. Hobbs, F. Martin, and R. Simmons. Conditions of learning in novice programmers. Journal of Educational Computing Research, 2(1):37-55, 1986.